Generalized Gromov Wasserstein Distance for Seed-Informed Network Alignment

Mengzhen Li[1] and Mehmet Koyutürk[1]

Department of Computer and Data Sciences,
Case Western Reserve University, Cleveland OH, USA
mxl994@case.edu, mxk331@case.edu

Abstract. Network alignment is a commonly encountered problem in many applications, where the objective is to match the nodes in different networks such that the incident edges of matched nodes are consistent. Gromov–Wasserstein (GW) distance, based on optimal transport, has been shown to be useful in assessing the topological (dis)similarity between two networks, as well as network alignment. In many practical applications of network alignment, there may be "seed" nodes with known matchings. However, GW distance assumes that no matchings are known. Here, we propose Generalized GW-based Network Alignment(GGWNA), with a loss/distance function that reflects the topological similarity of known matching nodes. We test the resulting framework using a large collection of real-world social networks. Our results show that, as compared to state-of-the-art network alignment algorithms, GGWNA can deliver more accurate alignment when the seed size is small. We also perform systematic simulation studies to characterize the performance of GGWNA as a function of seed size and noise, and find that GGWNA is more robust to noise as compared to competing algorithms. The implementation of GGWNA and the Supplementary Material can be found in https://github.com/Meng-zhen-Li/Generalized-GW.git.

Keywords: Gromov-Wasserstein Distance, Network Alignment

1 Introduction

Network alignment is the problem of aligning nodes that belong to the same entity from different networks based on the similarity of their connections [14]. In social networks, network alignment is often used to match the users that are the same person [12]. In biological networks, network alignment is used to identify molecules with similar evolutionary history and/or biological function [8].

Gromov-Wasserstein (GW) distance [9] is a measure that aims to quantify the distance between two networks (or similarity matrices) based on their topological (dis)-similarity. The formulation of GW derives an optimal transport (OT) [16], which compares probability distributions and minimizes the transport cost between the distributions [11]. There are many existing variations of GW distance. Entropic GW distance [10] introduces an entropic regularizer to

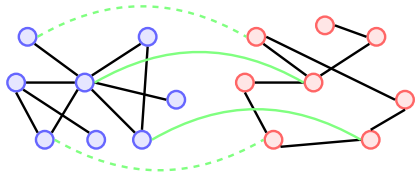


Fig. 1: **Illustration of the seed-informed network alignment problem.** Given the blue and red networks, and the known mappings of some nodes in the networks (solid green lines), the objective is to identify mappings of other nodes (dashed green lines) to maximize topological consistency.

the loss function. Sliced GW [15] projects each distribution in an 1D form and improves efficiency.

The computation of GW distance between two networks also entails computation of a fuzzy mapping (the "transport" matrix) between the nodes of the two networks, which is useful for network alignment. Motivated by this observation, many recent studies develop GW-based methods for network alignment[2, 17]. GW is also shown to be useful in computing node embeddings for multiple networks, by jointly performing graph alignment and node embedding [17].

The classical formulation of GW distance and its existing variations assume that the mapping between the nodes of the two networks is unknown (or irrelevant) and formulate the optimization problem purely based on topology. However, in graph alignment applications involving real-world networks, there are some known matchings (Figure 1), which can be used as prior knowledge in computing the mapping of remaining node pairs [4]. In this paper, we propose a novel framework for Gromov-Wasserstein based network alignment and introduce a new loss function that takes into account the known matchings between the two networks as "seed nodes" used to guide the alignment process. The proposed "generalized Gromov-Wasserstein distance" fixes the known matching of seed nodes in the optimal transport, while incorporating the topological consistency of these nodes in the loss function. We comprehensively assess the performance of the proposed Generalized Gromov-Wasserstein-based Network Alignment (GGWNA), in comparison to standard GW-based alignment, as well as other network alignment algorithms[18, 5]) on a rich corpus of social networks and synthetic datasets. We also investigate the effect of several factors and hyperparameters on the performance of GGWNA and other algorithms: 1) the number of seed nodes that are available, 2) the node overlap between the networks, 3) the divergence of edges between the two networks, and 4) the relative importance assigned to he topological consistency between seed vs. free matchings in our loss function. Our results show that (i) the use of seed matchings greatly improves the accuracy of GW-based alignment, (ii) GGWNA performs better when more attention is given to the topological consistency of the seed nodes, (iii) GGWNA is drastically more robust than non-GW based algorithms to small seed sizes and more divergence between the networks. These results establish GW-based algorithms as a compelling alternative for seed-driven network alignment, while also enabling computation of GW distance for a broader range of networks.

2 Background

2.1 Optimal Transport

Optimal transport [10] minimizes the mapping cost between two probability distributions. Suppose that $p \in {\mathbb{R}_+}^m$ and $q \in {\mathbb{R}_+}^n$ are two distributions, given a cost matrix $C_{ij} \in \mathbb{R}^{m \times n}$ representing the transport cost from i to j. The optimal transport problem aims to find a matrix T to minimize the transport cost:

$$\textbf{minimize} \sum_{i,j} C_{i,j} T_{i,j} \quad \textbf{subject to:} \; T \in \mathbb{R}_+^{m \times n} : T \mathbb{1}_m = p, \; T^T \mathbb{1}_n = q. \quad (1)$$

This optimization problem can be solved using quadratic optimization[13]. In our experiments, we compute the optimal transport using the Python Optimal Transport (POT) package [3].

2.2 Gromov-Wasserstein Distance

Based on the optimal transport theory, GW distance [9] was proposed as a measure to quantify the (dis)similarity between two matrices. GW distance is defined between (C_1, p) and (C_2, q), where C_1 and C_2 are two similarity matrices that represent the pairwise similarities or distances of elements, p and q are the two distributions that represent the relative importance of the elements [10].

This representation can be applied to quantifying the (topological) dissimilarity between two networks G_1 and G_2, as $C_1 \in \mathbb{R}^{m \times m}$ and $C_2 \in \mathbb{R}^{n \times n}$ can be selected as the adjacency matrices of G_1 and G_2. In its most general setting, the GW distance between two adjacency matrices C_1 and C_2 is defined as:

$$GW(C_1, C_2, p, q) = \min_T \sum_{i,j,k,l} L(C_1(i,k), C_2(j,l)) T(i,j) T(k,l) \quad (2)$$

where i and k refer to nodes in G_1, j and l refer to nodes in G_2, p and q are vectors representing the relative importance of the nodes in the two networks, $L(.)$ is a loss function, and T is constrained by p and q as in (1). In common applications of Gromov-Wasserstein based network distance, quadratic loss $L(a,b) = \frac{1}{2}|a-b|^2$ is used along with uniform distributions for p and q, i.e., $p = \frac{1}{m}\mathbb{1}_m$ and $1 = \frac{1}{n}\mathbb{1}_n$ [10].

2.3 Network Alignment Problem

Network alignment aims to find a matching between the nodes of two networks, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ to maximize the consistency of the incident edges of matched nodes. Network alignment algorithms differ in terms of how they formulate an objective function to reflect this aim, as well as how they solve the resulting optimization problem(s) [5, 6, 18]. Network alignment algorithms can be supervised [18] or unsupervised [5]. GW-based network alignment formulates the problem as an optimization problem as in (2), where T represents

the resulting mapping of the nodes. Here, we consider the seeded variant of the problem, where the matching between a subset of nodes $S = V_1 \cap V_2$ is known. The objective of seeded network alignment is to find a mapping between the nodes in $V_1 - S$ and $V_2 - S$ to maximize topological consistency.

3 Methods

3.1 Generalized Gromov-Wasserstein with Known Matching Nodes

Suppose that we have two networks $G_1 = \{V_1, E_1\}$ and $G_2 = \{V_2, E_2\}$ such that a subset of nodes $S = V_1 \cap V_2$ is common. We aim to compute T, $|V_1 - S| \times |V_2 - S|$ transport matrix such that $T(i, j)$ provides a mapping of the remaining nodes that maximizes topological consistency of the networks, given S.

Generalized Gromov-Wasserstein Distance. Let C_1 and C_2 denote the adjacency matrices of G_1 and G_2. Reorganize matrix C_i $(i = 1, 2)$ as follows:

$$\begin{array}{c} \\ S\left\{\vphantom{\begin{array}{c}A\\B\end{array}}\right. \end{array} \begin{array}{c} \overbrace{}^{S} \\ \left[\begin{array}{c|c} A_i & B_i \\ \hline B_i' & D_i \end{array}\right] \end{array}$$

Here, A_i corresponds to the edges between nodes that exist in both networks, B_i and B_i' correspond to edges between one node in S and one node outside S, and D_i corresponds to edges that are between nodes outside S. Since the mapping of nodes in S are fixed, the topological consistency of A_1 and A_2 is not informative on the mapping of the nodes in $V_1 - S$ vs. $V_2 - S$. Thus we consider the topological consistency of B_1 vs B_2, B_1' vs B_2', and D_1 vs D_2 to generalize Gromov-Wasserstein distance for this scenario:

$$L_1 = \sum_{\substack{i,k \in V_1 - S \\ j,l \in V_1 - S}} \frac{1}{2}(D_1(i,k) - D_2(j,l))^2 T(i,j) T(k,l) \tag{3}$$

$$L_2 = \sum_{\substack{i \in S \\ k \in V_1 - S \\ l \in V_1 - S}} \frac{1}{2}(B_1(i,k) - B_2(i,l))^2 T(k,l)^2 \tag{4}$$

Here, L_1 is the same as the GW distance between D_1 and D_2. L_2 considers each common node $i \in S$, and penalizes the inconsistencies in the neighborhood of i created by the mapping of other nodes in the networks. We define the generalized Gromov-Wasserstein distance as the weighted sum of these two loss functions:

$$L_{generalized} = \min((1 - \alpha)L_1 + \alpha L_2) \tag{5}$$

Here, $0 \leq \alpha \leq 1$ is a parameter that balances the relative importance of prior information (edges with one side fixed) vs. free mappings (edges with both sides to be mapped). Increasing α assigns more weight to L_2, so that the learning algorithm depends more on the known matchings instead of other nodes. $\alpha = 0$

corresponds to standard GW distance between D_1 and D_2 (ignoring the parts of G_1 and G_2 that are induced by the seeds), while $\alpha = 1$ corresponds to taking into account the edges incident to seeds only.

Peyré et al.[10] propose an efficient learning algorithm for computing the GW distance by incorporating a 4-way tensor $\mathcal{L}$ and a tensor matrix multiplication $\mathcal{L} \otimes T$. The loss function of Gromov-Wasserstein distance can be rewritten as:

$$GW(C_1, C_2, T) = \langle \mathcal{L}(C_1, C_2) \otimes T, T \rangle \tag{6}$$

in which $\mathcal{L}(C_1, C_2) \otimes T$ is the cost matrix C in the optimal transport.A decomposition of $\mathcal{L}(C_1, C_2) \otimes T$ is also proposed to improve efficiency.

The optimal transport can be computed by solving a quadratic optimization problem [13]. Building on this approach, we propose an efficient learning algorithm for computing the Generalized GW disance by generalizing the quadratic problem to fit our objective function. For this purpose, we first define a $|V_1 - S| \times |V_2 - S|$ matrix:

$$E(k, l) = \sum_{i \in S} (B_1(i, k) - B_2(i, l))^2 \tag{7}$$

which is a constant matrix and can be computed by matrix operations. Then the loss function of the generalized Gromov-Wasserstein becomes:

$$L_{generalized} = \langle (1 - \alpha)\mathcal{L}(D_1, D_2) \otimes T + \alpha E \odot T, T \rangle \tag{8}$$

where $E \odot T$ is the element-wise multiplication of E and T, and $(1-\alpha)\mathcal{L}(D_1, D_2) \otimes T + \alpha E \odot T$ is the cost matrix C.

We use Algorithm 1 to compute the Generalized GW distance of two networks. We first initialize the optimal transport T as the outer product of p and q (defined in Section 2.1). At each iteration, we compute the gradient direction of T and use Algorithm 2 to compute the optimal learning rate τ to minimize the cost of $T + \tau \Delta T$:

$$\tau = \arg \min_{0 \leq \tau \leq 1} L_{generalized}(T + \tau \Delta T) \tag{9}$$

The update function L_1 is derived in [13] as a quadratic function of τ. Using E as defined above, we derive the following update function for L_2:

$$L_2(B_1, B_2, T + \tau \Delta T) = \sum_{\substack{k \in V_1 - S \\ l \in V_1 - S}} E(k, l)(\Delta T(k, l)\tau^2 + 2T(k, l)\Delta T(k, l)\tau + T(k, l)^2) \tag{10}$$

Thus the update function for $L_{generalized}$ can also be expressed as a quadratic function of τ. We then compute the optimal learning rate τ as the value that minimizes the resulting update function for $L_{generalized}$ and update T accordingly. When $\tau = 0$ or ΔT is less than a threshold, the process converges and stops. The complexity of the algorithm is $O(mn^2 + m^2 n)$, where m and n are the number of nodes in $V_1 - S$ and $V_2 - S$.

Algorithm 1 Optimization for GGWNA

$T^{(0)} \leftarrow pq^T$
for i = 1, 2, ... **do**
  $C \leftarrow$ cost matrix of the iteration
  $T \leftarrow OT(C, T^{(i-1)})$
  $\Delta T \leftarrow T - T^{(i-1)}$
  $\tau^{(i)} \leftarrow$ line search using algorithm 2
  $T^{(i)} \leftarrow T^{(i-1)} + \tau^{(i)} \Delta T$
end for

Algorithm 2 Line Search

$a \leftarrow -2(1-\alpha)\langle D_1 \Delta T D_2, \Delta T\rangle + \alpha\langle E \odot T^{(i-1)}, T^{(i-1)}\rangle$
$b \leftarrow (1-\alpha)\langle c_{D_1,D_2}, \Delta T\rangle$ - $2(1-\alpha)(\langle D_1 \Delta T D_2, T^{(i-1)}\rangle + \langle D_1 T^{(i-1)} D_2, \Delta T\rangle) + 2\alpha\langle E \odot T^{(i-1)}, \Delta T\rangle$

$c \leftarrow L_{gw}(T)$
if $a > 0$ **then**
  $\tau \leftarrow \min(1, \max(0, \frac{-b}{2a}))$
else
  $\tau \leftarrow 1$ if $a + b \leq 0$ else $\tau \leftarrow 0$
end if

Algorithm 3 Greedy Matching

Require: optimal transport $T \in \mathbb{R}^{p \times q}$
Ensure: array $M \in \mathbb{R}^{\min(p,q) \times 2}$ of matchings
$M \leftarrow \varnothing$
while size of $M < \min(p, q)$ **do**
  $i, j \leftarrow$ the row and column indices of $\max(T)$
  if $i \notin M(1)$ and $j \notin M(2)$ **then**
    add node pair $[i, j]$ to M
  end if
end while

3.2 Seeded Network Alignment Using Optimal Transport

Having computed the optimal transport matrix T, we aim to find an optimal matching between the two networks. For a pair of nodes $i \in G_1$ and $j \in G_2$, T_{ij} is assigned a larger value by the optimal transport algorithm if the local topology around them are more similar (also considering their edges with the nodes in S). While there are many algorithms in the literature to compute a discrete mapping of the nodes based on the weights in T [1], these algorithms are computationally costly. Here, since our focus is on computing T (as opposed to using T to compute a mapping), we use a simple greedy algorithm (Algorithm 3) to compute a mapping T, thereby enabling repeated computational experiments to compare the proposed algorithm against alternative algorithms. The framework we propose here can be used with any matching algorithm once T

Table 1: **The networks used in the experiments.** Left: Real network pairs. Right: Networks used to create network pairs in simulation studies .

Network Pairs	#Nodes	#Edges	#Matchings
Douban Offline	1118	1511	1118
Douban Online	3906	8164	
ACM	9872	39561	6325
DBLP	9916	44808	
Twitter	5120	130575	1609
Foursquare	5313	54233	
Phone	1000	41191	1000
Email	1003	4627	

Networks	#Nodes	#Edges
Facebook	4039	88234
lastfm	7624	27806
Arxiv	5242	14496

is computed using Algorithm 1. In each iteration of this algorithm, we find the row and column indices of the maximum value in T, and align the corresponding pair of nodes. If one of the nodes is already aligned, we skip the pair and find the next maximum value in T, until $\min(|V_1 - S|, |V_2 - S|)$ nodes are aligned.

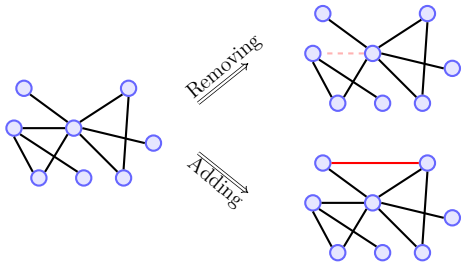
4 Experimental Results

4.1 Datasets

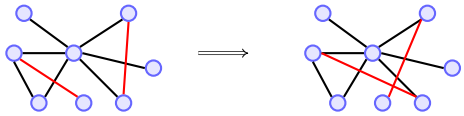
We use real-world social network pairs to compare GGWNA with other network alignment algorithms. The network pairs [18] used in our experiments are shown in Table 1. Douban is an online social network providing user review and recommendation services for movies, books, and music. ACM and DBLP are two co-authorship networks, in which nodes indicate authors and edges indicate that the two authors published at least one paper together. The twitter-foursquare data includes friend relationships from two online social networks, and the overlaps are the people who are in both networks. In the Phone-Email dataset [19], the Phone and Email networks respectively correspond to communications among people via phone and emails. For all datasets, the matchings are the users that are identified as the same person in different social networks.

Besides real-world network pairs, we perform simulation studies on real-world network[7] to generate synthetic network pairs with controlled characteristics. We assess the effect of the following variables in simulation studies(figure 2):

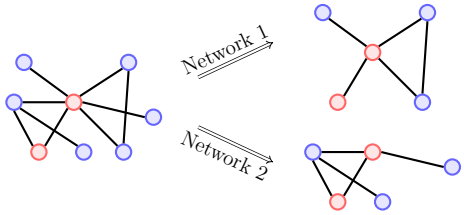
- **Network pairs with different levels of divergence:** For each network $G = (V, E)$ in Table 1, we generate 10 networks by adding or removing $\gamma|E|$ edges from G, where γ represents divergence (also referred to as noise, varying from 0.05 to 0.8). In the experiments, we align the 10 new networks with the original network G and assess the mean and variance of accuracy.
- **Divergent network pairs with identical degree distribution:** For each network $G = (V, E)$ in table 1, we generate 10 divergent networks with γ: 0.05, 0.1, 0.2, 0.4, 0.8. To preserve degree distribution, we randomly remove two randomly selected edges $(i, j), (k, l) \in E$, and add edges (i, k) and (j, l) at each iteration of the randomization process (repeated $\gamma|E|/2$ times).



(a) Adding noise by adding/removing randomly selected edges. Left: Original network. Upper right: Dashed edge is removed. Lower right: Red edge is added to the network.



(b) Adding noise by swapping nodes in two randomly selected edges. Left: Original network. Right: Network after one edge swap.



(c) Splitting a network into two networks with fixed overlap. The red nodes selected from the original graph are the overlapping nodes of the new graphs.

Fig. 2: **Simulation techniques used to generate synthetic network pairs.**

– **Network pairs with different levels of node overlap:** We simulate the case when two partial observations of a network are aligned. We generate 10 network pairs with different levels of node overlap: 0.1, 0.2, 0.4, 0.8. For a network $G = (V, E)$ in Table 1(Right), we split it into two networks, where $\lambda|V|$ (λ denotes the overlap parameter) nodes appear in both networks, and other nodes are equally distributed in the two networks. If there is an edge $(i, j) \in E$, then edge (i, j) also appears in the new networks. After the pair is constructed, we add 20% noise to both networks as described above.

4.2 Baseline Methods

GW: The Gromov-Wasserstein distance was introduced in Section 2.2. We learn the optimal transport matrix using all nodes (including seed nodes) to apply the greedy matching algorithm, ignoring the seed matching.

FINAL [18] is a supervised network alignment method for attributed networks. The FINAL algorithm leverages the node/edge attribute information to guide topology-based alignment process. In our experiments, the networks are not attributed networks, so the node attribute matrices are empty, and only topological consistencies are considered. We use the default hyperparameters of FINAL.

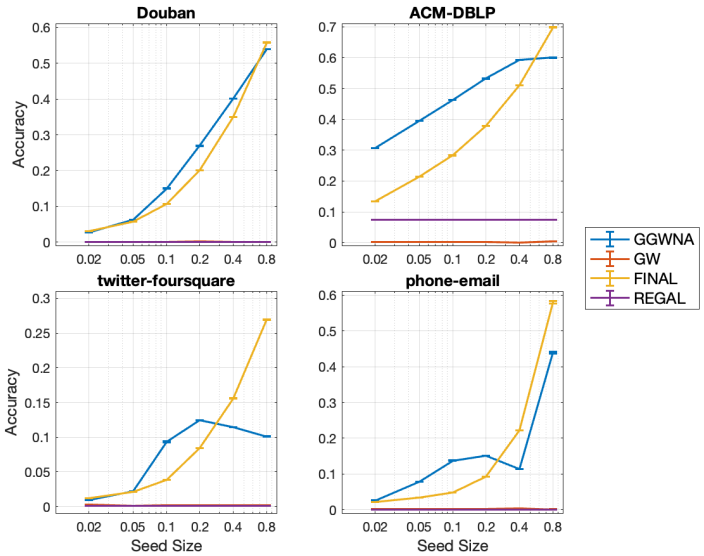


Fig. 3: **Comparison of network alignment algorithms on real-world network pairs.** The x-axis shows the percent of overlap that are used to train the models. The curves and error bars show the means and variances across 10 runs. Here, $\alpha = 0.8$ for GGWNA (the effect of this parameter is shown in Figure 4).

REGAL [5] first learns a node embedding for each network by a proposed matrix factorization technique (xNetMF). Then, the embeddings are used to compute the cross-network node similarities of each pair of nodes.

4.3 Experimental Setup

We compare our method with the baseline methods in terms of the accuracy of network alignment. Let $S' = V_1 \cap V_2$ denote the set of all known matching in the two networks. For a given "seed size" σ (fraction of known matchings in the training set), we randomly select $\sigma|S'|$ nodes from S' to construct S. The remaining nodes in $S' - S$ become the test test. For all algorithms S is provided as the set of seed matching and the resulting mapping of the node pairs in $S' - S$ is obtained by using the greedy matching algorithm (Algorithm 3) on the weighted mapping matrix returned by the algorithm. The accuracy is of alignment is then computed as the fraction of correctly aligned pairs in $S' - S$. All the experiments are repeated 10 times, and the averages and variances are shown in the figures. The x-axes of all figures are in log scale.

4.4 Results on Real Network Pairs

The network alignment accuracy of the four algorithms on four different pairs of real network pairs as a function of seed size is shown in Figure 3. On all datasets, GGWNA and FINAL clearly outperform GW and REGAL. In addition, for all datasets, GGWNA outperforms FINAL for smaller seed sizes, while FINA

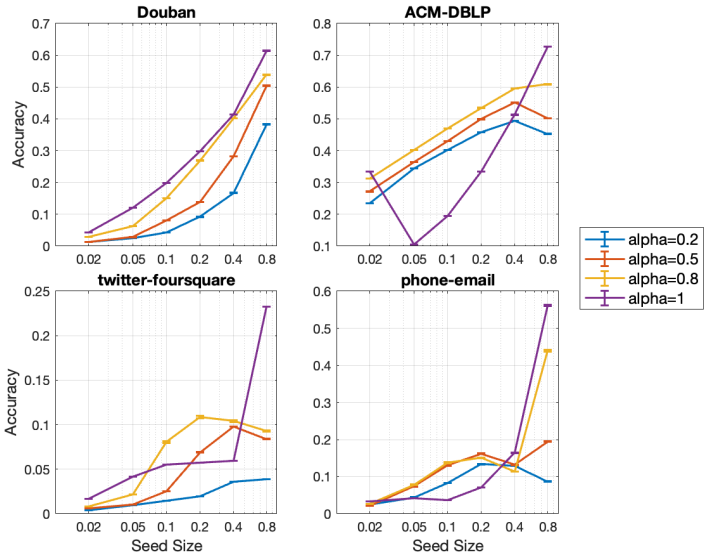


Fig. 4: **The effect of α on the performance of GGWNA.** The x-axis shows percent overlap used to train the models and the error bars show the variances.

outperforms GGWNA when the seed size is large. These results suggest that GGWNA is quite robust to smaller seed size.

In most cases, the accuracy of GGWNA increase as the seed size becomes larger, but the performance of GGWNA begins to decline when the seed size is too large (40% or 80%). The reason might be: As the seed size becomes larger, the D_i part of the adjacency matrices (Section 3.1) is smaller and will have less weight in the loss function. Therefore, the learning process depends more on the topological features of the seed nodes and less on the topological similarities of the nodes in the test set. FINAL performs better than GGWNA when more seed nodes are used in training, but 80% seed sizes can be unrealistic in practice. GW and REGAL do not work well on these datasets, and since they are unsupervised, the accuracy does not increase as the seed size becomes larger.

Figure 4 shows the effect of α on the performance of GGWNA. Overall, the accuracies of GGWNA increases as α increases from 0.2 to 0.8, but the performance goes down in most cases as we increase α to 1, since we depend too much on the known matchings instead of the topology. As α goes higher, the weight of the known matchings increases, and the optimal transport will depend more on B_1 and B_2 parts of the adjacency matrices.

4.5 Results on Simulated Pairs of Networks

We investigate the effect of various parameters using simulated network pairs(figure 2). We show the results on networks generated using the Facebook dataset here, the results of other datasets are in the Supplementary Material.

The Effect of Divergence/Noise. As seen in Figure 5, the accuracy of the algorithms declines as the two networks diverge. GGWNA is most robust against

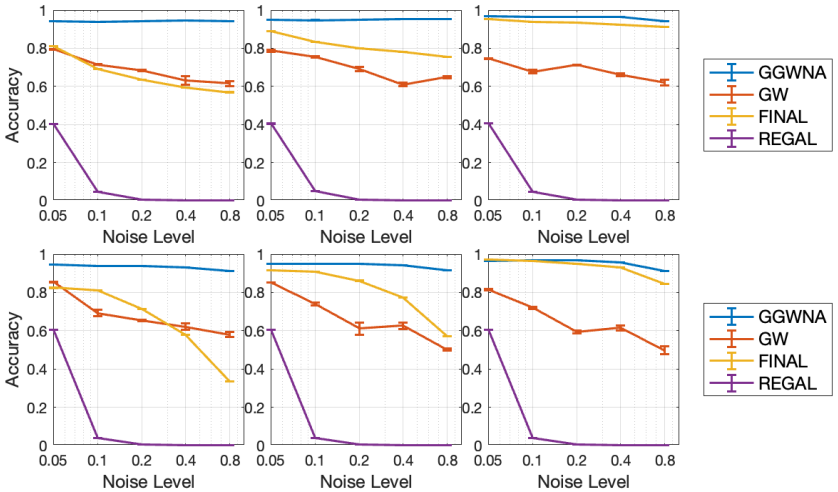


Fig. 5: **Accuracy of network alignment accuracy as a function of noise/divergence between networks.** Top: Uniform noise, Bottom: Degree-preserving noise. The seed sizes are 10% (left), 20% (center), and 50% (right).

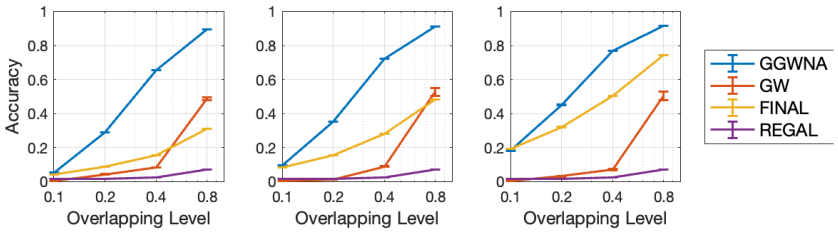


Fig. 6: **Network alignment accuracy on partial observations of a network.** Two networks are samples from the original network, with overlap levels from 10% to 80% as shown on the x-axis. 20% random noise is added to each network. Seed size: 10% (left), 20% (center), 50%(right) of the overlapping nodes.

noise, the accuracy decreases slightly as the noise level increases. The accuracy of FINAL improves as the seed size increases from 0.05 to 0.8, but GGWNA remains at a higher accuracy even when the seed size is small. Accuracy declines more sharply for degree-preserving noise (bottom panel), since this presents a more difficult instance for the algorithms (i.e., the algorithms cannot use node degree information to match the nodes), which can be more relevant in practice.

Partial Observations of a Network. From the results on real network pairs (Figure 3), we observe that GGWNA works better than other techniques when the node overlap between nodes the networks is larger (e.g. the ACM-DBLP and phone-email datasets). In the experimens reported in Figure 6, we investigate the effect of node overlap between two observations of a single network. As seen in the figure, the accuracy of all algorithms improves as the node overlap becomes larger, especially for GGWNA. However, GGWNA is still robust to smaller seed sizes as there is no obvious differences between the curves of the three subplots.

5 Conclusions

In this paper, we proposed generalized Gromov-Wasserstein for network alignment (GGWNA), by introducing a new loss function that takes into account the connectives of seed nodes for which matchings are known. We compared the accuracy of the algorithm on real network pairs as well as simulated pairs, and showed that our generalized GW outperforms other network alignment methods at most time, and it is robust to high divergence between networks and smaller seed sizes. Avenues for future research include introducing labels into the loss function, and applying generalized GW to a broader range of types of networks.

References

1. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE TPAMI **31**(6), 1048–1058 (2009)
2. Chowdhury, S., Needham, T.: Generalized spectral clustering via gromov-wasserstein learning. In: ICAIS, pp. 712–720 (2021)
3. Flamary, R.: Pot: Python optimal transport. JMLR **22**(78), 1–8 (2021)
4. Halimi, A., Ayday, E.: Profile matching across online social networks. In: Information and Communications Security: 22nd International Conference, ICICS 2020, Copenhagen, Denmark, August 24–26, 2020, Proceedings 22, pp. 54–70. Springer (2020)
5. Heimann, M., Shen, H., Safavi, T., Koutra, D.: Regal: Representation learning-based graph alignment. In: ACM CIKM, pp. 117–126 (2018)
6. Kazemi, E.: Proper: global protein interaction network alignment through percolation matching. BMCBbioinformatics **17**(1), 1–16 (2016)
7. Leskovec, J.: Snap datasets: Stanford large network dataset collection (2014)
8. Ma, L.: Heuristics and metaheuristics for biological network alignment: A review. Neurocomputing (2022)
9. Mémoli, F.: Gromov–wasserstein distances and the metric approach to object matching. Foundations of computational mathematics **11**(4), 417–487 (2011)
10. Peyré, G., Cuturi, M., Solomon, J.: Gromov-wasserstein averaging of kernel and distance matrices. In: ICML, pp. 2664–2672 (2016)
11. Santambrogio, F.: Optimal transport for applied mathematicians. Birkäuser, NY **55**(58-63), 94 (2015)
12. Shu, K., Wang, S., Tang, J., Zafarani, R., Liu, H.: User identity linkage across online social networks: A review. Acm SIGKDD Explorations **18**(2), 5–17 (2017)
13. Titouan, V., Courty, N., Tavenard, R., Flamary, R.: Optimal transport for structured data with application on graphs. In: ICML, pp. 6275–6284 (2019)
14. Trung, H.T.: A comparative study on network alignment techniques. Expert Systems with Applications **140**, 112,883 (2020)
15. Vayer, T., Flamary, R., Tavenard, R., Chapel, L., Courty, N.: Sliced gromov-wasserstein. arXiv preprint arXiv:1905.10124 (2019)
16. Villani, C.: Topics in optimal transportation, vol. 58. AMS (2021)
17. Xu, H., Luo, D., Zha, H., Duke, L.C.: Gromov-wasserstein learning for graph matching and node embedding. In: ICML, pp. 6932–6941 (2019)
18. Zhang, S., Tong, H.: Final: Fast attributed network alignment. In: ACM SIGKDD, pp. 1345–1354 (2016)
19. Zhang, S., Tong, H., Jin, L., Xia, Y., Guo, Y.: Balancing consistency and disparity in network alignment. In: ACM SIGKDD, pp. 2212–2222 (2021)